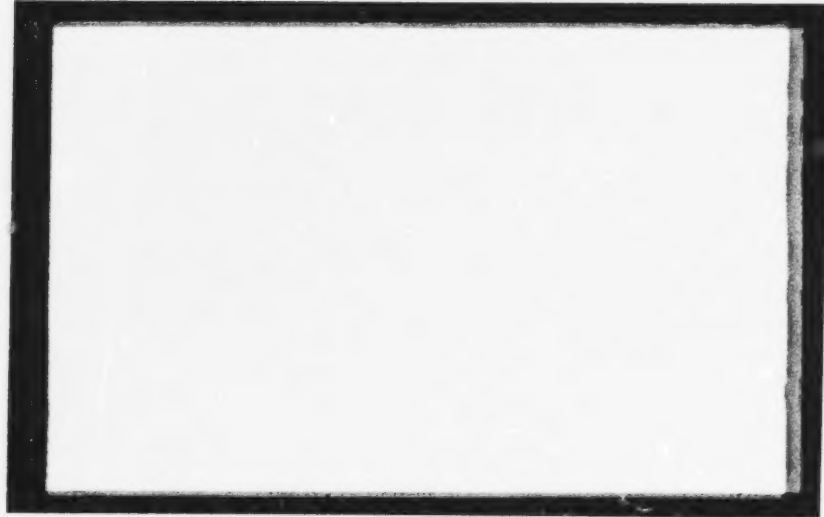


UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS



CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

**A BRANCH AND BOUND ALGORITHM FOR THE
CAPACITATED VEHICLE ROUTING PROBLEM**

by

Gilbert Laporte* and Yves Nobert**

*** Ecole des Hautes Etudes Commerciales de Montréal**

**** Département des Sciences administratives, Université
du Québec à Montréal**

The authors wish to thank the Canadian Natural Sciences and Engineering Research Council (grant A4747) and the Quebec Government (F.C.A.C. program) for their financial support. Thanks are also due to Martin Desrochers who solved the test problems.

Université de Montréal

**Centre de recherche sur les transports - Publication #283
Ecole des Hautes Etudes Commerciales -**

Cahiers du GERAD G-82-08

Septembre 1982

Dépôt légal - 1er trimestre 1983

1- INTRODUCTION

Problems of vehicle routing and scheduling hold a central place in physical distribution management. Operational researchers have devoted considerable effort to developing algorithms for their solution. One such problem concerns the optimal routing of vehicles of a given capacity through a set of customers (or cities) with known requirements; we shall refer to this as the capacitated vehicle routing problem (CVRP). More specifically, we consider

- (i) a set N of cities ($N = \{1, \dots, n\}$);
- (ii) a depot (city 1) used as the starting and terminating point for all vehicle trips (or routes);
- (iii) requirements (or weights) $d_1, \dots, d_n$ for all cities except the depot ($d_1 = 0$ and $d_2, \dots, d_n \geq 0$);
- (iv) a fleet of vehicles, each having a known capacity D ;
- (v) a matrix of distances $C = (c_{ij})$ between all pairs of cities; C is symmetrical if and only if $c_{ij} = c_{ji}$ for all $i, j \in N$; it is Euclidean if and only if $c_{ij} \leq c_{ik} + c_{kj}$ for all $i, j, k \in N$.

The CVRP consists of establishing vehicle routes such that

- (i) each city (except the depot) is visited exactly once;
- (ii) all routes start and end at the depot;
- (iii) the capacity of a vehicle on any given route is never exceeded, i.e. if S is the set of cities on such a route, then its weight $\sum_{i \in S} d_i$ does not exceed D ;
- (iv) the total routing cost is minimized.

ABSTRACT

This paper considers a version of the vehicle routing problem in which a non-negative weight is assigned to each city to be visited and where all vehicles are identical and have the same capacity D . The weight assigned to a vehicle on a given route may not exceed this capacity. The problem is formulated as an integer program: integrality is obtained by means of a branch and bound procedure; capacity constraints are first relaxed, and introduced only when they are found to be violated. Three variants of this basic algorithm are examined. Exact solutions are obtained for problems ranging from 15 to 50 cities.

In this problem, the number m of vehicles used in the solution can either be *fixed* (i.e. it is determined a priori) or *free* (i.e. its value is established simultaneously with the optimal routing).

Several related versions of the CVRP have been studied in the Operational Research literature. The most common variants include the following:

- (i) vehicles possess different characteristics regarding cost, capacity, speed, etc. (see [4], [6], [17] for recent surveys);
- (ii) the length of each route is constrained by an upper bound L [24];
- (iii) each city must be visited within a prescribed time interval [8];
- (iv) a combination of the above.

Our problem is therefore rather more specific, but nonetheless surprisingly difficult to solve optimally. The literature contains very few exact algorithms for the CVRP and none of them has been applied to problems involving more than 25 cities [4]. This compares with 325 cities for the asymmetrical travelling salesman problem (TSP) [1], 120 cities for the symmetrical TSP [19] and 400-500 cities for the non-Euclidean symmetrical multiple TSP (m-TSP) [16].

Many authors have developed formulations and/or heuristics for the CVRP. Comparisons of their contributions are difficult to make since various versions of the CVRP are treated, some results are valid for asymmetrical problems only (or for symmetrical problems only), and no measure of the departure from optimality is given. However, their work is generally appropriate for the approximate solution of

large scale and fairly complex problems. Moreover, the ideas they have developed have in some instances found their way into the construction of exact algorithms. As this paper is devoted to the elaboration of an exact algorithm for the CVRP, these contributions are not immediately relevant for our purpose. For more details concerning these methods, see [2], [9], [11], [13-15], [17], [27].

The main exact algorithms for the CVRP have been developed by Christofides [5] and by Christofides, Mingozzi and Toth [7].

Christofides' method [5] is based on the generation of feasible routes by means of a LIFO tree. The largest problem solved by this method contained 31 cities (this problem contained upper bound constraints on the route lengths). This algorithm is appropriate for tightly constrained problems as fewer feasible routes are then generated. More recently, Christofides, Mingozzi and Toth [7] have suggested algorithms for the case where m is fixed and where the length of each route is bounded above. The algorithms differ in the way lower bounds for the values of the subproblems (in a branch and bound tree) are generated. In one case, the computation of the bound is based on the notion of an m -tree also used by Held and Karp [20] for the TSP and by Gavish and Srikanth [16] for the m -TSP. In the second case, the bound is derived from the computation of q -paths, i.e. paths of the form $(1 - i_1 - \dots - i_k)$ where $\sum_{\ell=1}^k d_{i_\ell} \leq q$. (Note: the same city may appear more than once on a q -path). According to the authors, the bound derived from the q -paths is superior to the first one. These algorithms have been tested on 10 problems ranging from 10 to 25 cities and containing only capacity constraints.

More recently, Fleischmann [12] studied a version of the CVRP (called the VSP) in which only a subset N' of the cities have to be visited and where vehicles are allowed to go through the same city more than once. The problems studied by Fleischmann are derived

from "real life" road networks and the distance matrix is thus generally fairly sparse. Solutions for the VSP are obtained by solving optimally by linear programming a relaxed version of this problem called the VSP'. Computational results were reported for 6 problems in which $|N'|$ ranged from 28 to 68. In 4 cases, a feasible and optimal solution for the VSP was obtained.

Finally, note that vehicle routing problems of up to 60 cities, without capacity constraints but with an upper bound on the length of any route have been solved optimally by Laporte et al. [24]. As we shall see, the CVRP treated in this paper appears to be more difficult.

2- FORMULATION

In this paper, we have chosen to concentrate on symmetrical CVRP's. In addition to the notation already introduced, we define

f : a non-negative fixed cost on the use of a vehicle

$$\bar{S} = N - S \quad (S \subseteq N)$$

$$[t] = \begin{cases} \text{the smallest integer greater than or equal to} \\ t & \text{if } t \geq 0 \\ 1 & \text{if } t < 0 \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if a vehicle is used to travel directly} \\ & \text{between } i \text{ and } j \text{ } (i, j \in N - \{1\}). \\ 2 & \text{if a vehicle is used on a return trip} \\ & \text{between } i=1 \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

(Note: as the problem is symmetrical, x_{ij} is only defined for $i \leq j$; it is also assumed that $D \geq d_i$ ($i = 1, \dots, n$))

The problem can be formulated as follows:

$$(P) \quad \text{minimize} \quad z = \sum_{i < j} c_{ij} x_{ij} + fm$$

subject to

$$(1) \quad \sum_{j=2}^n x_{1j} = 2m$$

$$(2) \quad \sum_{i:k} x_{ik} + \sum_{j:k} x_{kj} = 2 \quad (k = 2, \dots, n)$$

$$(3) \quad \sum_{\substack{i,j \in N \\ i < j}} x_{ij} \leq |S| - \left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil \quad (S \subseteq N - \{1\})$$

$$(4) \quad \begin{aligned} x_{1j} &= 0, 1, 2 & (j = 2, \dots, n) \\ x_{ij} &= 0, 1 & (2 \leq i < j \leq n) \\ m &\geq 0 \end{aligned}$$

In this formulation, m can be a constant (m fixed) or a variable (m free). Constraints (1) and (2) are referred to as the degree constraints: they specify the degree of each node. Constraints (3) prohibit the formation of illegal subtours, i.e. subtours disconnected from the depot or whose total weight exceeds D . They constitute a generalization of the subtour elimination constraints set up by Dantzig et al. [10] for the TSP. Their validity can be established as follows: consider a set $S \subseteq N - \{1\}$. The total weight of all cities included in S is equal to $\sum_{i \in S} d_i$. Since each vehicle has a capacity D , the number of vehicles required to visit all points of S in the optimal solution will be greater than or equal to $\left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil$.

Since each vehicle will have to enter and leave S , it follows that

$$(5) \quad \sum_{\substack{i \in S, j \in \bar{S} \\ \text{or } i \in \bar{S}, j \in S}} x_{ij} \geq 2 \left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil$$

is a valid constraint for the CVRP. When (1) and (2) are satisfied, the equivalence between constraints (3) and (5) can be established in a fairly straightforward way [27].

It is obvious from the preceding discussion, that in problems where m is free, its value in the optimal solution will be bounded below by $\left\lceil \frac{\sum_{i \in N} d_i}{D} \right\rceil$. The following result provides an upper bound for m .

Lemma: Assume that m is free and that $c_{1i} + c_{1j} \geq c_{ij}$ for all i, j such that $2 \leq i < j \leq n$; then $\left\lceil \frac{2 \sum_{i \in N} d_i}{D} - 1 \right\rceil$ is a valid upper bound for m in the optimal solution.

Proof: First observe that the optimal solution contains at most one route whose weight does not exceed $\frac{1}{2}D$. Indeed, if there were two such routes, they could be amalgamated into a single feasible route without penalty (routes $1 - i_1 - \dots - i_k - 1$ and $1 - j_1 - \dots - j_\ell - 1$ where $i_k < j_1$ could be replaced by $1 - i_1 - \dots - i_k - j_1 - \dots - j_\ell - 1$ since $c_{1i_k} + c_{1j_1} \geq c_{i_k j_1}$). Similarly, if $D \leq \sum_{i \in N} d_i$, it is optimal to use only one vehicle and the conclusion holds trivially. Now assume that $D < \sum_{i \in N} d_i$. Let w_r be the weight of the r^{th} route in the optimal solution. Two cases are possible.

Case 1: $w_r > \frac{1}{2}D$ ($r = 1, \dots, m$). Then

$$(6) \quad \frac{1}{2}mD \leq \sum_{r=1}^m w_r = \sum_{i \in N} d_i$$

Case 2: $w_r \geq \frac{1}{2}D$ ($r = 1, \dots, m-1$) and $w_m < \frac{1}{2}D$.

Let $\delta = \min_{r=1, \dots, m-1} \{w_r - \frac{1}{2}D\}$ and $\varepsilon = \frac{1}{2}D - w_m$.

We must have $\varepsilon < \delta$ for otherwise, route m could be amalgamated with the route yielding δ . We can write successively

$$(7) \quad (m-1) \left(\frac{1}{2}D + \delta \right) + \left(\frac{1}{2}D - \varepsilon \right) \leq \sum_{r=1}^m w_r = \sum_{i \in N} d_i$$

$$(8) \quad m \leq 2(\delta + \varepsilon + \sum_{i \in N} d_i) / (D + 2\delta)$$

$$(9) \quad m < (2 \sum_{i \in N} d_i) / D$$

since $\varepsilon < \delta$ and $D < \sum_{i \in N} d_i$. The conclusion follows from (6) and (9). ■

Note that our formulation for the CVRP extends quite naturally to asymmetrical problems. (The number of x_{ij} variables and of degree constraints is then doubled.) In this case, it is possible to show that our formulation can be derived by applying Benders' decomposition [3] to the formulation given by Gavish and Graves [15] for the asymmetrical CVRP. The proof is rather lengthy and is provided fully in [27]. (Gavish and Graves have shown that a similar relationship exists between their model for the TSP [15] and that of Dantzig et al. [10]).

3- BASIC ALGORITHM

The formulation given in the previous section presents two major difficulties: the presence of a large number of integer variables and of an exponential number of subtour elimination constraints. In order to circumvent these difficulties, we propose

an algorithm in which integrality is reached via a branch and bound scheme of the type developed by Land and Powell [21] and in which type (3) constraints are initially relaxed and only introduced as they are found to be violated. A similar procedure has already been used successfully for other types of complex combinatorial problems [12], [23-27]. In such problems, integrality can also be attained by the imposition of Gomory cutting planes [18], but preliminary tests have shown clearly that in the case of the CVRP, too many cuts had to be generated: the memory was insufficient and the determinant of the basis soon became excessive, leading to unacceptable inaccuracies in the computations.

We describe in this section the basic branch and bound algorithm set up for the CVRP. (Some details have voluntarily been left out in order to simplify its presentation.) In the next section, three variants of the basic algorithm are introduced.

Step 0 : Initialization. If m is free, solve (P) by the Clarke and Wright algorithm [9]; if m is fixed, use the Fisher and Jaikumar method [11]. Let z^* be the value of the objective function of the best solution found. (Tests have shown that the Clarke and Wright algorithm is more appropriate to problems where each route contains relatively few cities whereas the Fisher and Jaikumar method is more efficient in the opposite situation).

Step 1 : Define a first subproblem consisting of the objective function, of constraints (1) and (2) and of the upper bounds for the variables. The level of the search tree is set equal to 0.

Step 2 : Identify the next problem to be solved according to the criterion defined in [21].

Step 3 : Solve the current subproblem by means of the simplex method. Let $\bar{z}$ be the optimal value of the objective function. (If the subproblem is infeasible, then $\bar{z} = \infty$).

Step 4 : If $\bar{z} = z^*$, the algorithm proceeds to step 10 as no better solution than the one available can be obtained by exploring further the current branch of the search tree.

Step 5 : Eliminate from the problem all ineffective subtour elimination constraints (if any have been generated). This operation is carried out by a subroutine derived from PURGE [21] and described in [27]. In principle, the removal of such constraints is not absolutely essential. It is however a very useful time and memory saving device for the ensuing subproblems. Extensive computational experience has shown that the probability of generating more than once the same subtour elimination constraint is in fact very slight. Therefore, the benefits of executing this step seem to outweigh its disadvantages.

Step 6 : If not all variables are integer, proceed to step 12.

Step 7 : An integer solution whose total cost is less than z^* has been identified. A check is made to find out whether it contains any illegal subtours. If so, step 8 is executed. Otherwise, the algorithm proceeds to step 9.

Step 8 : For each illegal subtour identified in step 7, the appropriate subtour elimination constraint is generated. Note that such constraints do not only eliminate the current subtours, but all other subtours that could result from a permutation of the cities they contain. The algorithm then proceeds to step 3.

Step 9 : A feasible solution better than the best one so far found has been obtained. It is stored and the value of z^* is updated.

Step 10 : The backtracking procedure described in [21] is executed. The level of the tree is reduced.

Step 11 : The candidate subproblems are updated as a branching or a backtracking procedure has just taken place. The algorithm then proceeds to step 13.

Step 12 : The current solution contains at least one fractional variable and $\bar{z} < z^*$. A variable on which to branch (x_t , say) is determined and a branching procedure is applied, i.e. new subproblems with $x_t = 0$, $x_t = 1$ (and possibly $x_t = 2$) are created and the level of the branch and bound tree is increased. The algorithm proceeds to step 11.

Step 13 : If the list of candidate subproblems is empty, the optimal solution has been found. Otherwise, the algorithm proceeds to step 2.

4- VARIANTS OF THE BASIC ALGORITHM

Three major variants of the algorithm described in the previous section have been tested.

Variant 1: generation of Gomory cuts at level 0

The first variant consists in strengthening the problem by introducing up to five Gomory cutting planes in the first subproblem. (These cuts are never "purged" in the algorithm.) The number five was determined empirically after it was found that the marginal effect of any more cuts was negligible. The main effect of these cuts is to increase $\bar{z}$ at an early stage and thus to restrict the growth of the search tree.

Variant 2: subtour prevention constraints

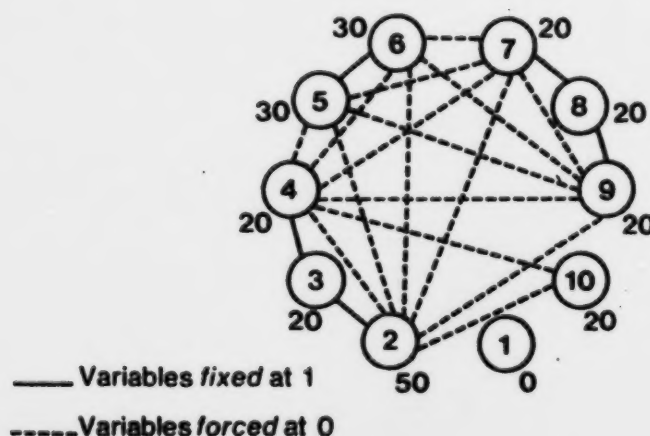
Under some conditions, it is possible to increase the efficiency of the algorithm by *preventing* the occurrence of illegal subtours rather than eliminating them, thus avoiding the unnecessary generation of many subtour elimination constraints. In order to illustrate this point, we consider the following example. Let $n = 10$, $D = 100$, $d_2 = 50$, $d_3 = d_4 = d_7 = d_8 = d_9 = d_{10} = 20$, $d_5 = d_6 = 30$ and assume that at some stage in the search tree, variables $x_{2,3}$, $x_{3,4}$, $x_{5,6}$, $x_{7,8}$ and $x_{8,9}$ have been fixed at 1.

It is clear in this case that city 10 cannot be on the same route as cities 2, 3 and 4, hence it is possible to set $x_{2,10} = x_{4,10} = 0$.

Similarly the chains (2-3-4) and (5-6), (2-3-4) and (7-8-9), (5-6) and (7-8-9) cannot belong to the same route. Hence, all the variables associated with edges linking the end points of these chains can also be forced at zero (see figure 1). The following subtour prevention constraint can then be imposed:

$$(10) \quad \begin{aligned} & x_{2,4} + x_{2,5} + x_{2,6} + x_{2,7} + x_{2,9} + x_{2,10} + x_{4,5} + x_{4,6} \\ & + x_{4,7} + x_{4,9} + x_{4,10} + x_{5,7} + x_{5,9} + x_{6,7} + x_{6,9} + x_{7,9} \leq 0 \end{aligned}$$

FIGURE 1: SUBTOUR PREVENTION CONSTRAINTS



It is therefore possible, when creating a new branch with $x_t = 1$ (step 12), to identify all potential infeasibilities resulting from the amalgamation of chains or isolated nodes on the same route. This is more efficient than eliminating these infeasibilities after they occur. When backtracking (step 10) in the search tree (from level L_1 to level L_2 , say), it is necessary to eliminate all subtour prevention constraints that have been generated at all levels between L_2 and L_1 , since the validity of these constraints is linked to the fact that some variables have been fixed at 1 between levels L_2 and L_1 . Ineffective constraints are "purged" according to the procedure described in [27]; effective constraints are first made ineffective by setting their right hand side equal to an appropriately large number and then by reoptimizing (see [22, chapter 7] for more details).

Variant 3: alternative branching strategy

Two branching strategies have been tested. The first one is quite standard and is used for example in the Land-Powell code [21]; it consists of selecting the (fractional) variable which would produce the largest increase in the value of the objective function if it was branched upon in a given direction, and then branching in the opposite direction. In the second strategy, variables are first fixed at 1 in order to facilitate the generation of as many subtour prevention constraints as possible.

Computational experiments with the three variants described in this section indicate that it is always advantageous to generate Gomory cuts at the first node of the search tree and to impose subtour prevention constraints whenever possible. These two options have therefore been used in all tests that follow. The same experiments do not give any clear edge to any of the two branching strategies. We have opted for the Land-Powell criterion which seems marginally better.

5- COMPUTATIONAL RESULTS

The algorithm presented in the previous sections was tested on a series of problems of various types and sizes on the University of Montreal CYBER 173 computer, using an FTN5 compiler.

In the case of Euclidean problems, n cities were first randomly located in $[0,100]^2$ and c_{ij} was defined as the Euclidean distance between the location of city i and that of city j . The c_{ij} coefficients for non-Euclidean problems and the weights d_i for both types of problems were randomly generated from a uniform distribution on $[0,100]$. In all cases, the c_{ij} 's and the d_i 's were rounded up or down to the nearest integer.

The generation of the vehicle capacity D is quite critical since its value has a great impact on the efficiency of the algorithm. First observe that D must be at least equal to $\max_{i=1,\dots,n} \{d_i\}$, since

otherwise the problem is infeasible. Moreover, if $D \geq \sum_{i=1}^n d_i$, the capacity constraints are always satisfied and the problem reduces to an m -TSP. We have therefore chosen to define D as

$$(11) \quad D = (1 - \alpha) \max_{i=1,\dots,n} \{d_i\} + \alpha \sum_{i=1}^n d_i$$

where α is a parameter chosen in the interval $[0,1]$. (Note that $\alpha = 1$ yields an m -TSP.) Since our algorithm proceeds by constraint relaxation, the hardest problems will be those for which α is small since then, more constraints will have to be generated. In an algorithm of the type described in [7], selecting a small value for α produces easier problems since in that case, fewer variables need to be generated; conversely, this type of algorithm is likely to

perform poorly in problems where α is relatively large. The type of algorithm best suited to a given problem can therefore be selected according to the value of α defined as

$$(12) \quad \alpha = (D - \max_{i=1, \dots, n} \{d_i\}) / (\sum_{i=1}^n d_i - \max_{i=1, \dots, n} \{d_i\}).$$

In the tests that follow, we have let α take the values 0, 0.25, 0.50, 0.75 and 1.

We first present in tables 1 and 2, results for Euclidean and non-Euclidean problems for values of n ranging from 15 to 50. Here, m was free and there was no fixed cost on the use of a vehicle. All results are average values for the number of successful problems out of 5 that were attempted (problems "failed" when they could not be solved within 300 seconds). As noted elsewhere [23-27], Euclidean problems are harder to solve than non-Euclidean problems. The largest non-trivial ($\alpha < 1$) problem solved contained 40 cities in the first case and 50 cities in the second case. As expected, the number of iterations and computation times became smaller as α increased. Since our algorithm is of the branch and bound type, the total number of constraints generated (and sometimes deleted) in the search tree is not really meaningful. Rather, we have recorded the maximum number of effective constraints (equivalent to the "maximum size of the inverse" in [21]) as a more significant measure. Since generally most or all of constraints (1) and (2) are effective in the optimal solution, the maximum number of effective subtour elimination constraints can be approximated by subtracting n from the maximum number of effective constraints. It can be seen that this number increases as n becomes larger but decreases as α approaches 1.

In table 3, we compare the impact on solution times of using three types of objective functions:

- (i) m free and $f = 0$;

- (ii) m free and $f = 10^6$; in this case the algorithm produces a solution with the minimum number of vehicles and having the smallest routing cost for that number of vehicles;
- (iii) m fixed at three values (2, 4 and 6) and $f = 0$.

Cases (i) and (ii) seem similar, with a slight edge in favour of m free and $f = 10^6$. Problems with m fixed appear on the whole to be easier than problems with m free, this becoming more so as m increases. A similar phenomenon was noted in the case of the multiple travelling salesman problem (see [27, chapter 3]) but, surprisingly, not in the distance constrained vehicle routing problem [24].

Finally, we present in table 4 ratios measuring the departure from optimality of the solutions produced by the Clarke and Wright [9] and by the Fisher and Jaikumar [11] algorithms, for some problems involving 15 or 20 cities and for $\alpha = 0.50$ or 0.75 . It can be seen that neither of these two algorithms produces a good solution in the case of non-Euclidean problems. On the other hand, the Clarke and Wright algorithm appears to be quite good in the case of Euclidean problems; the quality of the Fisher and Jaikumar method is also better in this case, but tends to worsen as m and n get larger.

TABLE 1

Euclidean problems - m free and $f = 0$

n	α	number of successful problems out of 5	number of simplex iterations	maximum number of effective constraints in the course of the algorithm	time (seconds)
15	0.00	3	1093.7	26.0	59.76
	0.25	3	930.3	28.7	74.32
	0.50	5	249.0	26.2	23.99
	0.75	5	70.0	22.6	4.17
	1.00	5	46.6	18.4	1.60
20	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	5	621.2	33.2	119.39
	0.75	5	172.8	28.8	21.38
	1.00	5	118.4	24.0	7.17
25	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	3	513.3	36.3	110.31
	0.75	5	290.8	33.4	57.13
	1.00	5	152.6	29.4	16.19
30	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	0	-	-	-
	0.75	3	582.7	41.3	133.03
	1.00	5	225.6	37.2	37.81
35	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	0	-	-	-
	0.75	1	119.0	37.0	18.16
	1.00	5	182.6	40.4	37.72
40	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	1	176.0	52.0	57.48
	0.75	1	168.0	53.0	50.25
	1.00	5	408.8	49.0	121.87

Non-Euclidean problems - m free and $f = 0$

17.

n	α	number of successful problems out of 5	number of simplex iterations	maximum number of effective constraints in the course of the algorithm	time (seconds)
15	0.00	4	1409.0	25.3	81.94
	0.25	2	1373.0	25.0	100.33
	0.50	5	118.8	22.0	7.48
	0.75	5	44.8	18.4	1.88
	1.00	5	38.0	16.6	1.15
20	0.00	1	2140.0	33.0	175.54
	0.25	3	1052.7	31.0	167.95
	0.50	5	398.4	28.6	55.56
	0.75	5	87.2	24.6	7.96
	1.00	5	65.6	23.2	4.66
25	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	3	189.7	31.3	29.78
	0.75	5	105.8	28.6	12.63
	1.00	5	111.2	27.2	12.12
30	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	1	114.0	33.0	14.22
	0.75	5	114.4	34.2	17.56
	1.00	5	95.4	31.8	11.52
35	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	2	93.5	38.0	15.95
	0.75	5	121.4	39.8	26.89
	1.00	5	123.2	36.6	23.45
40	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	3	175.0	46.3	117.73
	0.75	5	217.0	44.8	66.98
	1.00	5	151.0	42.2	41.35
45	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	2	194.5	52.0	162.73
	0.75	5	185.4	50.0	84.31
	1.00	5	223.4	48.8	96.47
50	0.00	0	-	-	-
	0.25	0	-	-	-
	0.50	0	-	-	-
	0.75	3	214.7	53.3	91.56
	1.00	5	179.4	51.6	68.36

TABLE 3

Comparison of three objective functions - average time in seconds
(5 successful problems out of 5 in each case)

Type of problems	n	α	m free f = 0	m free f = 10^6	m = 2 f = 0	m = 4 f = 0	m = 6 f = 0
Euclidean	15	0.50	23.99	17.61	19.78	3.55	0.92
		0.75	4.17	4.25	4.05	1.68	0.69
	20	0.50	119.39	91.96	53.35	35.15	22.15
		0.75	21.38	19.17	21.54	13.64	2.64
Non-Euclidean	15	0.50	7.48	7.17	7.12	1.52	0.86
		0.75	1.88	1.83	1.82	1.02	0.84
	20	0.50	55.56	51.49	46.68	4.80	7.59
		0.75	7.96	7.29	7.19	7.68	2.48

TABLE 4

Heuristic vs exact algorithms - average ratios
(5 successful problems out of 5 in each case)

Type of problems	n	α	CW/opt (1)	FJ/opt (2)		
			m free (f=0)	m = 2	m = 4	m = 6
Euclidean	15	0.50	1.02	1.02	1.27	1.43
		0.75	1.03	1.07	1.37	1.45
	20	0.50	1.08	1.04	1.39	1.49
		0.75	1.74	1.11	1.46	1.58
Non-Euclidean	15	0.50	1.37	1.35	1.48	1.51
		0.75	1.28	1.38	1.50	1.51
	20	0.50	1.53	1.45	1.85	1.63
		0.75	1.43	1.50	1.86	1.67

- (1) Local optimum (Clarke and Wright [9] algorithm) + Global optimum (our algorithm).
 (2) Local optimum (Fisher and Jaikumar [11] algorithm) + Global optimum (our algorithm).

BIBLIOGRAPHY

- [1] BALAS, E., CHRISTOFIDES, N.: "A Restricted Lagrangean Approach to the Travelling Salesman Problem", Mathematical Programming, vol.21, pp. 19-46, 1981.
- [2] BALINSKI, M., QUANDT, R.: "On an Integer Program for a Delivery Problem", Operations Research, vol.12, pp. 300-304, 1964.
- [3] BENDERS, J.F.: "Partitioning Procedures for Solving Mixed-Variables Programming Problems", Numerische Mathematik, Vol.4, pp. 238-252, 1962.
- [4] BODIN, L.D., GOLDEN, B.L., ASSAD, A., BALL, M.: "The State of the Art in Routing and Scheduling of Vehicles and Crews", Report No. UMTA/BMGT/MSS #81-001, U.S. Department of Transportation, Washington D.C., 396 pp., 1981.
- [5] CHRISTOFIDES, N.: "The Vehicle Routing Problem", RAIRO (recherche opérationnelle), Vol.10, pp. 55-70, 1976.
- [6] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "The Vehicle Routing Problem", in Combinatorial Optimization, pp. 315-338, Wiley, 1979.
- [7] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Mathematical Programming, Vol.20, pp. 255-282, 1981.
- [8] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems", Networks, Vol.11, pp. 145-164, 1981.
- [9] CLARKE, G., WRIGHT, J.W.: "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", Operations Research, Vol.12, pp. 568-581, 1964.
- [10] DANTZIG, G.B., FULKERSON, R., JOHNSON, S.: "Solution of a Large-Scale Travelling-Salesman Problem", Operations Research, Vol.2, pp. 393-410, 1954.
- [11] FISHER, M., JAIKUMAR, R.: "A Generalized Assignment Heuristic for Vehicle Routing", Networks, Vol.11, pp. 109-124, 1981.

- [12] FLFISCHMANN, B.: "Linear Programming Approaches to Travelling Salesman and Vehicle Scheduling Problems", Paper presented at the XI. International Symposium on Mathematical Programming, 20 pp., Bonn, August 1982.
- [13] FOSTER, B.A., RYAN, D.M.: "An Integer Programming Approach to the Vehicle Scheduling Problem", Operational Research Quarterly, Vol.27, pp. 367-384, 1976.
- [14] GARVIN, W.M., CRANDALL, H.W., JOHN, J.B., SPELLMAN, R.A.: "Applications of Linear Programming in the Oil Industry", Management Science, Vol.3, pp. 407-430, 1957.
- [15] GAVISH, B., GRAVES, S.C.: "The Travelling Salesman Problem and Related Problems", Working Paper No. 7905, Graduate School of Management, University of Rochester, 32 pp., 1979.
- [16] GAVISH, B., SRIKANTH, K.: "An Optimal Solution Method for the Multiple Travelling Salesman Problem", Working Paper No. 8027, Graduate School of Management, University of Rochester, 43 pp., 1982.
- [17] GOLDEN, B., MAGNANTI, T.L., NGUYEN, H.Q.: "Implementing Vehicle Routing Algorithms", Networks, Vol.7, pp. 113-148, 1977.
- [18] GOMORY, R.: "An Algorithm for Integer Solutions to Linear Programs", Recent Advances in Mathematical Programming, pp. 269-302, McGraw-Hill, 1963.
- [19] GROTSCHER, M.: "On the Symmetric Travelling Salesman Problem: Solution of a 120-City Problem", Mathematical Programming Study, Vol.12, pp. 61-77, 1980.
- [20] HELD, M., KARP, R.M.: "The Travelling Salesman Problem and Minimum Spanning Trees: Part II", Mathematical Programming, Vol.1, pp. 6-25, 1971.
- [21] LAND, A.H., POWELL, S.: "FORTRAN Codes for Mathematical Programming: Linear, Quadratic and Discrete", Wiley, 249 pp., 1973.
- [22] LAPORTE, G.: "Permutation Programming: Problems, Methods and Applications", Ph.D. Thesis, University of London, 358 pp., 1975.

- [23] LAPORTE, G., NOBERT, Y.: "An Exact Algorithm for Minimizing Routing and Operating Costs in Depot Location", European Journal of Operational Research, Vol.6, pp. 224-226, 1981.
- [24] LAPORTE, G., NOBERT, Y., DESROCHERS, M.: "Two Exact Algorithms for the Distance Constrained Vehicle Routing Problem", Cahiers du GERAD G-82-05, Ecole des Hautes Etudes Commerciales de Montréal, 19 pp., 1982.
- [25] MILIOTIS, P.: "Integer Programming Approaches to the Travelling Salesman Problem", Mathematical Programming, Vol.10, pp. 367-378, 1976.
- [26] MILIOTIS, P., LAPORTE, G., NOBERT, Y.: "Computational Comparison of Two Methods for Finding the Shortest Complete Cycle or Circuit in a Graph", RAIRO (recherche opérationnelle), Vol.15, pp. 233-239, 1981.
- [27] NOBERT, Y.: "Construction d'algorithmes optimaux pour des extensions au problème du voyageur de commerce", Thèse de doctorat, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 240 pp., 1982.

